University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Protein domain boundary prediction based on deep neural networks

Alexandros Angeli

March 24, 2023

Abstract

The study of proteins is essential for understanding biological processes and developing drugs and treatments for disease. However, experimental methods for determining protein structure can be expensive and time-consuming. In this project, we explore the use of a deep neural network to predict protein domain boundaries which provide structural information for proteins. In addition, we investigate the performance of the neural network when using different protein language models to encode amino acid sequences. We found out that our model is effective at classifying proteins as single-domain or multi-domain, but lacks performance in predicting the precise positions of domain boundaries in the sequence. Despite this limitation, our method provides valuable information about the structural properties of proteins. Additionally, we evaluated a novel 3D structure prediction method called AlphaFold for domain boundary prediction by translating its 3D structure predictions to domain boundaries. Our results provide insights into the performance of AlphaFold in the task.

Acknowledgements

This four-year journey would never have been possible without the support and sacrifices made by my parents. To Mum and Dad: thank you.

I would like to extend my thanks to my supervisor, Dr Kevin Bryson, for all the guidance, patience and expertise provided throughout my dissertation project, as well as for all the hours he spend on our meetings and on answering any questions I had.

Last but not least, a big thank you to my four-legged best friend Oscar, who has been a constant source of comfort and joy throughout my studies.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Alexandros Angeli Date: 24 March 2023

Contents

1 Introduction

1.1 Motivation

1.1.1 Protein structure

Proteins are essential molecules for living organisms. It is approximated that a human body contains between 80,000 and 400,000 such proteins (Watson 2021). They are involved in many different functions of the human body including, but not limited to, transporting oxygen in the blood, catalyzing chemical reactions and providing protection against pathogens. Each protein is made up of a unique sequence of amino acids. Various inexpensive and quick experimental methods have been devised over the years that can determine the amino acid sequence from the protein itself. The particular amino acid sequence determines the physiochemical properties of a protein which, in turn, determine how a protein folds and what 3D structure[1] it takes. The specific shape of a protein is related to its unique functionality.

Understanding the structure of proteins is of utmost importance. Designing drugs that can target particular proteins can aid drug discovery, engineering new proteins with desired properties can be used in biotechnology applications and understanding how proteins misfold and aggregate can help develop new treatments for diseases such as Parkinson's disease and Alzheimer's. Along with these, there are many other instances in which understanding a protein's structure can be of use.

Experimental methods for determining protein structures exist and can be highly accurate. However, they are associated with many challenges which provide an incentive to explore new methods for the task. Significant limitations of experimental methods include how time consuming and expensive they are. Additionally, some proteins are very difficult to study as they are too complex, hard to purify or too unstable. Therefore, new computational methods for protein structure prediction and understanding that can overcome these limitations are in high demand.

1.1.2 Protein domains

A protein domain is a unit of a protein that folds and functions independently from the protein. Figure 1.1 shows an example of the structure of a protein highlighting its three domains. A protein domain boundary is where two domains separate.

The identification of these boundaries enables the division of proteins into smaller, independent domains. Subsequently, other methods can be applied to these domains, such as in 3D structure prediction, which may lead to better results. Additionally, breaking down proteins into domains facilitates the process of crystallisation[2], which also aids in structure prediction using experimental methods.

[1] The atomic coordinates of a protein in 3D space.

[2] A step in a process called X-ray crystallography which is an experimental method for determining the 3D structure of a protein.

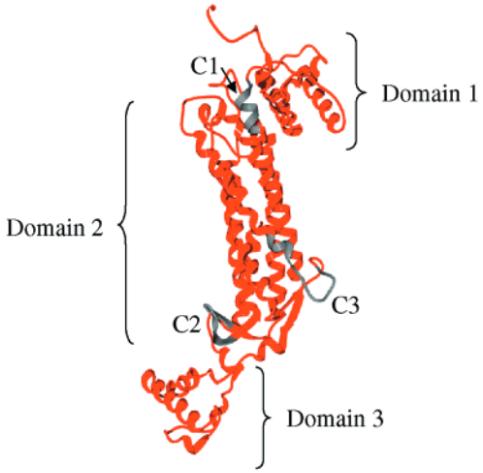


Figure 1.1: *A schematic diagram of the three dimensional topology of a protein indicating its three structural domains (Sampaleanu et al. 2001)*

1.2 Aims

1.2.1 Application of deep learning to domain boundary prediction

In light of the above motivations, we will explore how well a deep neural network can predict protein domain boundaries from the amino acid sequence. To achieve this we will:

- describe the problem of protein structure prediction from the amino acid sequence using machine learning based approaches.
- collect and process adequate data that will be used for training.
- implement and train a deep neural network with an architecture suited for the problem domain using different amino acid sequence encoding mechanisms.
- evaluate and analyse the model's results and compare them with other similar machine learning methods that exist in the literature.
- discuss the limitations of our methods and suggest possible improvements that could have been adopted in our solution.

1.2.2 Evaluation of AlphaFold for domain boundary prediction

In chapter 2 we introduce and discuss a novel algorithm for predicting the 3D structure of a protein from its amino acid sequence, called AlphaFold. In light of its success at the 14th Critical Assessment of Structure Prediction[3], we aim to provide an insight into its efficacy in protein domain boundary prediction. The motivation for doing so is to argue whether domain boundary prediction could be facilitated in its process to improve its predictions. Our goals are to:

- describe the problem of evaluating AlphaFold for the domain boundary prediction task.
- collect appropriate AlphaFold-predicted 3D structures and translate them to AlphaFold-predicted domain boundaries.
- evaluate the results and compare them with results obtained from methods devised for explicitly predicting domain boundaries, including our model.
- discuss our findings and the limitations of our process.

[3]Community wide experiment to determine and advance the state of the art in modeling protein structure from amino acid sequence

2 Background

2.1 Proteins

2.1.1 Amino acid sequence

Amino acids are organic compounds that form the building blocks of proteins. Each is made up of different atoms and therefore has different properties. There are hundreds of unique amino acids found in nature but only about 20 are needed to make all the proteins found in the human body and most other forms of life (Lopez and Mohiuddin 2022). Table 1 presents the names of the 20 amino acids and their abbreviations.

2.1.2 Chains

Inside proteins, amino acids are linked together by chemical bonds, known as peptide bonds, to form polypeptide chains. Once linked in the chain by a peptide bond, an amino acid is called a residue. Throughout the dissertation we will refer to polypeptide chains as simply "chains". Proteins can be made up of one or more chains.

2.1.3 Domain prediction and discontinuity

As described earlier, a protein domain is a unit of a protein that folds and functions independently from the protein. Most residues in the chain fall into a domain. However, residues can also form short sequences to separate domains. These residues are called linkers. Figure 2.1 shows an example of linkers separating two different domains.

Discontinuous domains are domains that contain more than one fragment from different regions of the sequence. Figure 2.1 shows an example of a discontinuous domain. When dealing with only continuous domains, domain boundary prediction and domain prediction are essentially synonymous as it is given that where there is a domain boundary there is the start of a new domain. However, in the presence of discontinuous domains the assumption that the number of domains N_d is equal to the number of domain boundaries $N_b + 1$ fails.

Discontinuity is a challenging problem in domain prediction, where the aim is to predict in which domain a residue falls into. This is evident from the fact that state-of-the-art methods in domain boundary prediction perform well in boundary prediction but either do not attempt or fail to correctly assign domains to residues.

2.2 Available data

2.2.1 RCSB Protein Data Bank

RCSB Protein Data Bank (PDB) is a publicly available archive of 3D structure data for large biological molecules, including proteins. Each entry is uniquely identified using a 4-character

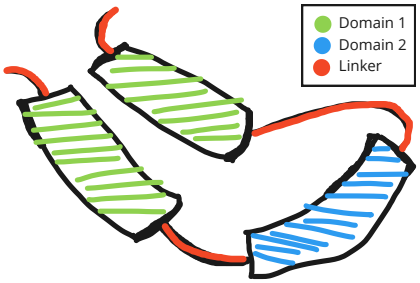


Figure 2.1: *An example of a discontinuous domain. The green domain is discontinuous, the blue domain is continuous. The orange lines are linkers. Even though the green domain is made up of two different regions in the residue sequence the chain folds in such a way where both regions are close to each other and therefore form one domain.*

PDB ID. For every[1] entry, there exists a `.PDB` file that contains the amino acid sequence for every chain in the protein as well as the atomic coordinates (the 3D structure) of the protein. The 3D structure of the protein has been determined using experimental methods, such as X-ray crystallography, which is the most accurate, and expensive, option available for determining structural information of proteins.

2.2.2 CATH

CATH is a publicly available online resource that provides information on the evolutionary relationships of protein domains. It provides a comprehensive list of protein chains coupled with information about their domains. Figure 2.2 provides an illustrative explanation of the format of the list.

- PDB ID
- Chain
- Domain
- Residue range

- 10gsA01 2-78,187-208
- 10gsA02 79-186
- 10gsB01 2-78,187-208
- 10gsB02 79-186

Figure 2.2: *A subset of the list mentioned in Section 2.2.2 provided by CATH, which provides information about protein domains. The first 4 characters indicate the PDB ID of the protein. Then follows the chain ID, as proteins can have more than one chain, and then the domain. Finally the numbers indicate the residue range that make up the domain. The information provided is interpreted as follows. The protein with PDB ID 10gs has two chains, A and B. Domain 1 of chain A is formed at positions 2 to 78 and then (an example of a discontinuous domain) from positions 187 to 208. Domain 2 of chain A makes up positions 79 to 186. Both chains are identical in terms of their domains.*

2.2.3 UniProt

UniProt is a database providing functional information for proteins such as descriptions of the known, or predicted, functions of proteins. The information is gathered from genome sequencing

[1] Except for exceptionally large and complex proteins.

projects. Each entry in the database is uniquely identified by an accession number. UniProt is independent from PDB and therefore a cross-reference to PDB is provided for every entry in UniProt. It is worth noting that, some proteins are very popular for medicine or biotechnology and as a result there exist more than one cross-reference to RCSB because of the number of different structures available for the proteins.

2.3 Convolutional neural networks

2.3.1 Convolutional layers

Convolutional layers is they key characteristic of convolutional neural networks. Unlike a fully connected layer, which connects all neurons in one layer to every neuron in the next layer, a convolutional layer only connects each neuron to a small, local region of the input. This allows for effective local feature learning. Convolutional neural networks are very popular in image processing tasks such as image classification, as convolutional layers are able to capture the characteristic features in an image or even an amino acid sequence.

2.3.2 Convolutional autoencoders

An autoencoder is a special type of neural network that consists of an encoder part and a decoder part, and learns a latent space of size z for an input of size x). The latent space is learned through the optimization of the model's parameters to minimize the reconstruction error between the input and output. The encoder maps the input to a compressed latent representation, while the decode reconstruct the original input from the latent representation, which is the output of the encoder. A convolutional autoencoder uses convolutional layers during the encoding and decoding stages. This technique can be useful in learning meaningful features, such as the features in images, or the features in long amino acid sequences. Furthermore, the use of convolutional layers during encoding can allow the autoencoder to capture relative positional information of the input.

2.4 Recurrent neural networks

2.4.1 Classic recurrent neural networks

Recurrent neural networks (RNNs) is a class of neural networks suited for processing sequential data. The name "recurrent" comes from the fact that these networks include cycles in their architecture, allowing them to perform recurrent computations while taking past information as input and incorporating it into their current output, making them attractive for sequence-based predictions. However, they suffer from a well-known problem called gradient vanishing[2], rendering them incapable to model long-term dependencies. This limitation presents a motivation to investigate alternative architectures.

2.4.2 Long short-term memory

Hochreiter and Schmidhuber (1996) present the long short-term memory (LSTM) which is a special type of a recurrent neural network capable of addressing the challenge of long-term dependencies by partially[3] solving the vanishing gradient problem. LSTMs are capable of capturing long-term interactions during learning by controlling the flow of information within the network. This feature makes LSTMs more suitable than RNNs for longer sequences of data.

[2] A problem in deep learning where the gradient of the loss with respect to the current weight becomes vanishingly small, preventing or completely stopping the neural network from further training.

[3] The gradient can still vanish but not as rapidly as it does in classic RNNs.

2.4.3 Bidirectionality

Bidirectional RNNs (Bi-RNNs) process a sequence of inputs in both forward and backward directions, enabling the network to learn from both past and future inputs concurrently. This addresses the limitation of RNNs, and in turn of LSTMs, which only incorporate past information into the current output, neglecting the potential usefulness of future information in prediction.

2.5 Sequence encoding

2.5.1 One-hot

One-hot encoding is a process for converting categorical data to binary vectors which can be used as training data in machine learning. Each binary vector is filled with zeros in all indices except the index that corresponds to the category being represented. In the case of converting amino acids into binary vectors, since there are 20 unique amino acids, each is represented by a binary vector of length 20 with zeros everywhere except the i_{th} position if the amino acid is the i_{th} amino acid in the list of unique amino acids.

2.5.2 Evolutionary Scale Modeling (ESM)

Rives et al. (2021) have trained an unsupervised deep neural network using the masked language modeling task (Devlin et al. 2018) to learn protein representations. An intuitive explanation as to why this task helps the network learn meaningful representations is that in order for the network to make a prediction about a masked amino acid, it needs to identify dependencies between the masked amino acid and the unmasked amino acids in the sequence. The network is based on the Transformer (Vaswani et al. 2017) architecture. It is claimed that the learned representations produce features that generalize across a range of applications including structure prediction. A number of pre-trained[4] models from this study are available to the public and have been used in recent protein domain boundary prediction methods (Wang et al. 2022).

2.5.3 Convolutional autoencoding representations of proteins (CARP)

Yang et al. (2022) suggest that while recent successful models for learning protein representations, like ESM, rely on the Transformer architecture which scales quadratically with sequence length in both run-time and memory, a convolutional autoencoder (Bank et al. 2020) may be more efficient. In response to this limitation, they designed and trained a convolutional autoencoder on the masked language modeling task (Devlin et al. 2018), which they demonstrate to be competitive or superior to Transformers in downstream applications such as structure prediction. Unlike Transformers, convolutional autoencoders use convolutions that scale linearly and are able to incorporate relative positional information by modeling sequences as sliding windows of amino acids. Despite its potential advantages, this recent study has yet to be applied to protein domain boundary prediction in existing literature.

2.6 Domain boundary prediction

The method of Postic et al. (2017) predicts domain boundaries from the experimental structures[5] of proteins. The method takes an approach that splits the structure into compact fragments and then test several possible domain delineations that are evaluated using the separation and the compactness. The definitions of this these criteria are beyond the scope of this dissertation.

[4]Models which their weights do not need to be optimised as the model has already been trained.
[5]The 3D structure as determined by experimental methods.

During evaluation, 87.7% of the residues were assigned to the correct domain. Furthermore, as their results showed to be close to results obtained when testing their method with more tolerance, it has been concluded that that finding the correct number of domains is a more challenging problem than delimiting accurate boundaries. This method goes one step beyond boundary prediction to predict the domain each residue belongs to. This means that discontinuous domains can be identified.

Jiang et al. (2019) was the first, to our knowledge, approach using deep learning to predict domain boundaries from the amino acid sequence alone. The method followed an *ab initio* approach without using third party methods - as seen in the studies below - which can introduce limitations such as in the length of the chain that can be taken as input. This method only depends on the amino acid sequence which is used to train a 4-layer stacked[6] bidirectional LSTM (Bi-LSTM). To train the neural network, each amino acid sequence is split into fragments using a sliding window, which are then input to the network. The output of the last layer is passed through a softmax activation function to assign a probability to each residue indicating whether it is a boundary, not a boundary or padding. Each residue was encoded using numerical descriptors collected from public databases. One limitation of this method is that the amino acid sequence is not encoded globally as breaking down a sequence into fragments prevent the LSTM from capturing long-term dependencies.

Shi et al. (2019) present a method called DNN-Dom for predicting domain boundaries and argue that local and non-local interactions between residues provide useful information for the overall structure of the protein. This is taken into account and a method that captures both local and non-local interactions has been devised using a convolutional neural network (CNN) in combination with a stacked bidirectional gated recurrent unit (Bi-GRU) (explain?). A third-party method, SCRATCH (Cheng et al. 2005), is used to extract two features from each chain, namely the secondary structure and solvent accessibility[7]. These two features are numerical values and are the input to the neural network, along with the Position-specific scoring matrix profile[8] for each chain. It is worth noting that SCRATCH accepts only chains of length shorter than 1500, which, in turn, limits DNN-Dom as well.

The work of Zheng et al. (2020) offers its own contribution in protein structure prediction by predicting the domain each residue belongs to which provides more information about the structure of the protein than the domain boundaries do.

Wang et al. (2022) offer one of the latest contributions to the field using a deep neural network which consists of several residual blocks[9] and a Bi-LSTM. The chains are encoded using ESM which has been discussed in subsection 2.5.2. In addition, they extract three more features from the amino acid sequence: solvent accessibility, secondary structure[10] and the hidden markov model (HMM) profile[11]. During their evaluation, two identical deep neural networks were trained with and without the features generated by ESM and it was concluded that when including the features in the model's input, the model yielded better results. Furthermore, the evaluation of the study shows that this method outperforms many state-of-the-art methods in the literature when tested on independent data. However, this method suffers from the same limitation as DNN-Dom as it depends on SCRATCH to extract features from the sequences.

[6] The output of one LSTM is input to another LSTM creating a "stack" of LSTMs.

[7] The extent of accessible surface area of a given residue.

[8] A matrix that contains information about the evolutionary history of a given protein sequence.

[9] Multiple sequential layers in a neural network where the input of the first layer is added to the output of the last layer, essentially creating a "skip connection" in a network.

[10] The class of the local structure of a chain.

[11] A probabilistic model that captures the diversity of biological sequences.

2.7 AlphaFold

2.7.1 Algorithm

AlphaFold2 (AF2) (Jumper et al. 2021) is the most successful and accurate method to date for predicting the 3D structure of a protein according to the 14th Critical Assessment of Structural Prediction (Pereira et al. 2021). The network consists of two main components. The first is a Transformer model that generates contextual representations from the input data: a contact map, which captures which residues interact with each other, and a representation of the chain's homologous sequences[12]. The second component is responsible for generating the 3D structure of the protein in the form of a rotation and translation for each residue, utilizing the generated representations and an ensemble of multiple prediction models. Figure 2.3 illustrates the architecture and information flow in AF2. It is worth noting that, even though AF2 demonstrates promising capabilities, its practical use is under exploration, including protein domain boundary assignment.

2.7.2 AlphaFold Database

A database with AlphaFold-predicted protein structures in the form of a `.PDB` file has been made publicly available and is already used in drug discovery and vaccine development. Each entry is uniquely identified using an accession number, which is the same identifier used in UniProt entries.

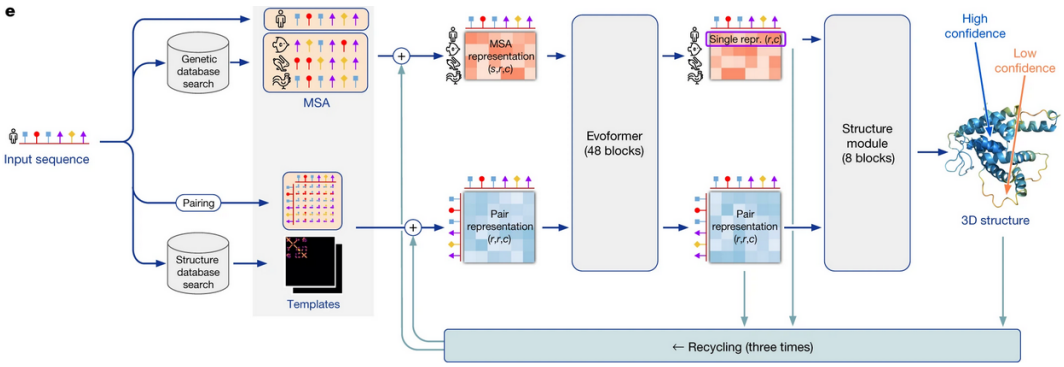


Figure 2.3: *An illustration of the architecture and information flow in AF2 (Jumper et al. 2021). Evoformer (the Transformer model) is the first component of the algorithm that generates contextual representations. The Structure Module is the second component of the algorithm that generates the 3D predictions.*

2.8 Summary

In this chapter we discussed what proteins are and defined the necessary terminology that will be mentioned throughout the dissertation. We introduced RNNs and RNN variants that solve technical challenges and improve learning. We introduced two recent and competitive methods for encoding amino acid sequences. We have presented related work in the field of protein domain boundary prediction that shows that non-experimental methods can be an effective and low-cost alternative at predicting structural information of proteins, yet, there is still room for improvement. Moreover, we discussed different deep learning approaches utilising different architectures and pointed out some of their limitations. Finally we introduced AF2, gave a

[12]Sequences with a common evolutionary origin.

high-level abstract explanation of its architecture and discussed its recent success in 3D structure prediction.

3 Analysis/Requirements

3.1 The thermodynamic hypothesis

The 3D structure of a protein determines the position of its domains, and therefore, determines the position of domain boundaries. This implies that if a factor determines the 3D structure of a protein, it also determines the position of domain boundaries. This factor is known as the thermodynamic hypothesis (Anfinsen 1973), which states that "the native [protein] structure is determined only by the protein's amino acid sequence". Therefore, we assume that there is a pattern in every amino acid sequence from which a machine learning algorithm can learn to predict domain boundaries. We will approach the problem of learning these patterns using deep neural networks that are able to learn complex, non-linear functions when given enough data.

3.2 Overcoming limitations of current methods

3.2.1 Sequence length

One limitation that stands out in the literature review in Chapter 2 is that, machine learning approaches, particularly Res-Dom and DNN-Dom, can only predict domain boundaries of amino acid sequences of a length shorter than 1500 residues. Considering that most amino acid sequences are between 50 and 2000 residues long (Alberts et al. 2002) and that a great incentive to use machine learning approaches to predict structural information of proteins is that some proteins are too long and complex to determine their structure, we are motivated to design a solution to the problem that is not limited by the sequence length.

3.2.2 Long-term dependencies

A further limitation, observed in the work of Jiang et al. (2019) is the inability to capture long term dependencies in the amino acid sequences, as each sequence is split into fragments that are then encoded and input to a neural network. This provides an incentive to design a solution to the problem that encodes the entire sequence, in order to utilise long-term dependencies in the sequence.

3.3 Using protein language models

The study of Wang et al. (2022) showed that ESM provides useful information when learning domain boundaries using deep neural networks, which suggests that pre-trained protein language models, such as ESM and CARP, can provide valuable information when using deep neural networks to learn domain boundaries. Motivated to further investigate the effectiveness of these models in the domain boundary prediction task, we will design a solution that will encode amino acid sequences using a pre-trained language model alone and compare it against a baseline solution.

3.4 Comparing CARP and ESM

After reviewing the literature in Chapter 2, it became apparent that there is lack of research that compares the performance of CARP and ESM, specifically in a controlled setting. Motivated by the recent publication of CARP and the claim that it is more efficient to train than alternatives that utilise the Transformer architecture, we wanted to explore how well it performs and how it compares to ESM.

3.5 Evaluating AlphaFold

We aim to evaluate the performance of AlphaFold in the task of domain boundary prediction and compare it both with the methods in the literature and with our solution. As AlphaFold is designed to predict the 3D structure of the protein from the sequence and not the domain boundaries, we must design a system to translate the 3D structure predictions into domain boundary predictions. By evaluating AlphaFold's performance, we can gain a better understanding of how it performs in this specific task and potentially identify any room for improvement by utilizing protein domain boundary information in the AlphaFold algorithm.

3.6 Summary

In this chapter, we described the problem of predicting protein domain boundaries from the amino acid sequence, and the process by which we arrived at this problem. We pointed out the limitations in the literature review from Chapter 2 and explained how we aim to design a solution that will overcome them. Furthermore, we explain the motive in using protein language models in our solution and identified a gap in the literature which is the lack of a quantitative comparison of two recent state-of-the-art protein language models. Finally, we discussed the problem of evaluating the performance of AlphaFold in the domain boundary prediction task and the motive to do so.

4 Methods

4.1 Training the neural network

4.1.1 Overview

The neural network will take sequences of amino acids of variable length L and will predict a probability vector of length L that assigns a value to every residue in the sequence, indicating the probability that the residue is a domain boundary. These probabilities will be computed by the neural network which learned what values yield the lowest error between the ground truth and the prediction. The error from a prediction will be passed to the neural network to adjust its parameters accordingly in order to improve its predictions.

4.1.2 Calculating the error

The ground truth of each sequence will be a binary vector of length L where every index i will have the value of 1 if the residue at position i is a domain boundary, otherwise the index will hold the value of 0. To calculate the error between the output by the neural network and the ground truth we will use an appropriate loss function to capture the discrepancy between two probabilities. Our choice of loss function is the binary cross entropy loss function, which is defined by:

$$L(\mathbf{X}) = \sum_{i=1}^{N} -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)), \tag{4.1}$$

where $\mathbf{X}$ is the input matrix to the neural network, y_i is the true classification of the i_{th} residue (or the probability that residue i is a boundary) and $\hat{y}_i$ is the predicted probability that the i_{th} residue is a boundary.

Each parameter in the neural network is optimised using the rate of change of the binary cross entropy loss with respect to itself using backpropagation. The goal is minimising this loss function, as the loss is always[1] positive and the loss closest to 0 is the most optimal.

4.2 Pre-processing

4.2.1 Input sequence

The machine learning algorithm cannot take a sequence of characters as input. Therefore, we will encode each sequence using an encoding mechanism that generates a representation of each sequence. The encoding mechanism will take an amino acid sequence of length L and will output a matrix of shape $L \times N$ where N is the number of features per residue. In Chapter 2 we introduced three different encoding mechanisms. We will use each independently and compare the performance of our neural network when using each of the encoding techniques in Chapter 5.

[1] Since the logarithm of a number less than 1 is negative, and the loss function is multiplied by -1.

4.2.2 Class imbalance

The number of boundaries in a sequence is disproportionate to the number of non-boundaries, as the most frequent number of boundaries is between 0 and 3 with sequences reaching a length of over 1000 residues. See Figure 4.6 for a visualisation of the distribution of sequence lengths in our dataset and Table 4.2 for the number of domains in the chains in our dataset. Due to the imbalance between boundaries and non-boundaries in every sequence, it will be very difficult for the neural network to learn effectively as it will be biased towards the majority class which is non-boundary. We can mitigate this challenge using two different approaches.

The first approach is to use a weighted loss function during training. The weighted binary cross entropy loss is defined by:

$$L(\mathbf{X}) = \sum_{i=1}^{N} -w_i \cdot (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)), \tag{4.2}$$

which is similar to Equation 4.1, except there is a w_i term, which is the inverse frequency of the class y_i in the binary vector. The effect of w during training the neural network is that a wrong prediction of the majority class will not be penalised as much as a wrong prediction of the minority class.

The challenge with this approach is the following. The weight of each class must be calculated for every batch, as the number of boundaries differs across batches, which can significantly increase the time required for the neural network to train. Additionally, if we choose to use a batch size greater than 1, then due to padding[2], the frequency of the non-boundary class will overestimated which will lead to the inverse frequency being smaller than optimal.

After performing hyperparameter tuning, we trained our model with each of the two approaches for one epoch and observed that when employing a weighted loss function the training time increased by 105% (from 192 seconds two 394), even when using vectorised operations for efficiency. In light of the limited available hardware and big difference in training time, we will not be considering this approach.

The second approach to mitigate the problem of class imbalance is to amplify the signal of each boundary by assigning neighboring residues of every boundary the value of 1, creating regions where a domain boundary is likely to exist and increasing the number of the boundary class in each sequence. We can achieve this amplification effect by performing a convolution between the binary vector (the ground truth) with a vector full of ones of size K, where K is the length of the boundary regions we wish to create. Consider an example where the binary vector is as follows:

$$\vec{y} = [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0].$$

The output of the convolution $\vec{\mathbf{y}} * \vec{\mathbf{h}}$, where $\vec{\mathbf{h}}$ is a vector of ones of size 3 is:

$$\vec{y} * \vec{h} = [0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0].$$

This approach will require to post-process the output from the model and choose one residue of every boundary region to be the domain boundary.

[2]Making every encoded sequence have the same size by appending a necessary number of zeros to each matrix row in order for every sequence to be of equal length, which allows the network to train in batches.

4.3 Post-processing

4.3.1 From regions to single boundaries

After amplifying the signal of the domain boundaries by creating regions where a domain boundary is likely to exist we must remove the regions and leave only one position per region as the domain boundary. The naive way to do this is to take the middle residue in every region of consecutive ones to be the domain boundary. A better approach is to take the residue with the highest probability in the region to be the domain boundary. In case where there is more than one position with the highest probability we will take the first one as the domain boundary. We will employ the latter approach during post processing.

4.3.2 Choice of cut-off threshold

In order to evaluate the performance of the neural network, we must convert the output probability vector into a binary vector. To do so, we will choose a threshold t such that probabilities above t will be classified as 1, otherwise they will classified as 0. Different values for t can yield different results. Therefore, we will optimize the choice of t by maximizing the Mathew's Correlation Coefficient (we discuss and justify our choice of metric to maximise in Chapter 5), which is a metric used to evaluate binary classifications. We employ the Algorithm 1 to optimise our choice of t.

Figure 4.1 shows how different values of t yield different results. This is particularly interesting because it shows how choosing different metrics to maximise leads to different optimal t values. For example, if we choose to optimise for the recall of the multi-domain class, then taking the value of 0 would yield the highest score, whereas optimising for the recall of the single-domain class, the value of 1 for t would yield the best result. The metrics are further discussed in chapter 5.

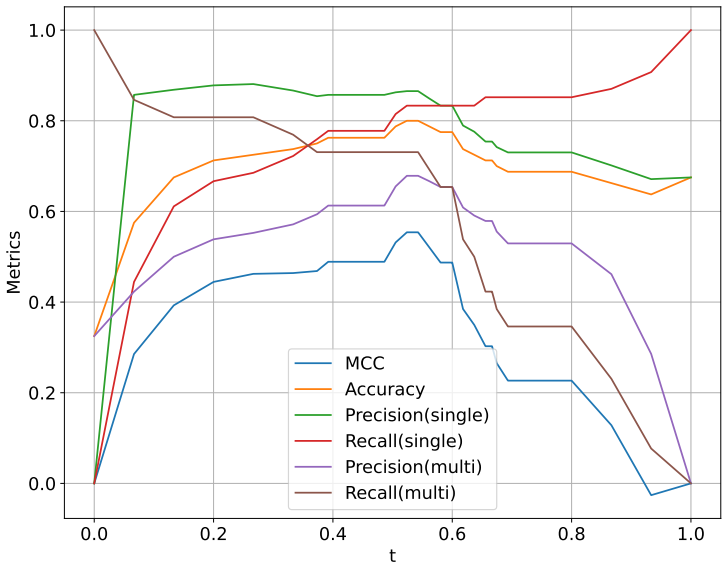


Figure 4.1: *The scores of our model during testing on an independent dataset plotted against the different values of t.*

Data: $start$, end The bounds of the possible values for t.
$has\_changed$, A flag indicating whether a better value for t has been found.
$max\_iterations$, The maximum number of iterations.
$step$, Determines by how much $start$ and end will change after every iteration.
n, The number of values for t per iteration.
m, The value to increase n by at every iteration.
Result: $highest\_score$, The highest MCC score obtained on the test data.
$best\_t$, The value for t that yielded that highest score.

```
begin
    best_score ⟵ 0
    best_t ⟵ 0
    start ⟵ 0
    end ⟵ 1
    has_changed ⟵ True
    n ⟵ 10
    m ⟵ 3
    while max_iterations > i and has_changed = True do
        has_changed ⟵ False
        range ⟵ EvenlySpacedValues(start, end, n) /* array of n evenly spaced values from start
          to end */
        for i = 0; i < n; i = i + 1 do
            score ⟵ MCC(t)
            if score > best_score then
                best_t ⟵ t
                best_score ⟵ score
                has_changed ⟵ True
            end
        end
        start ⟵ best_t · (1 − step)
        end ⟵ best_t · (1 + step)
        n ⟵ n + m
    end
end
```

Algorithm 1: The algorithm to find an optimal value for a cutoff t using the Mathew's Correlation Coefficient (MCC). The algorithm uses an iterative search to find the best threshold value for a classification problem. It calculates the MCC as a score using a range of evenly spaced values for for threshold t, and updates the best score and threshold value if a better score is found. The search continues until a maximum number of iterations or until no better score is found. The code adjusts the range of threshold values, decreasing the granularity of the values and repeats the process until convergence.

4.3.3 An alternative approach

A different approach of going from a proability vector to a binary vector is taking the highest probabilities in the probability vector as the domain boundaries. However this method requires to know beforehand how many domain boundaries exist in a chain. There is the approach of employing transfer learning[3] and train the network to learn how many domain boundaries there are in a given sequence and then produce a probability vector. However, this requires a change in the design of the neural network architecture as well as in the learning process and the choice of loss function. Due to limited resources we decided to not attempt this approach.

4.3.4 Overview of post-processing

During post-processing we will choose one position per boundary region as the domain boundary by selecting the position with the highest probability in the region as the domain boundary. Afterwards, we will convert the probability vector to a binary vector. To do this we will use a cutoff threshold t which will maximise the MCC of the model. Figure 4.2 provides an illustrative description of the post-processing.

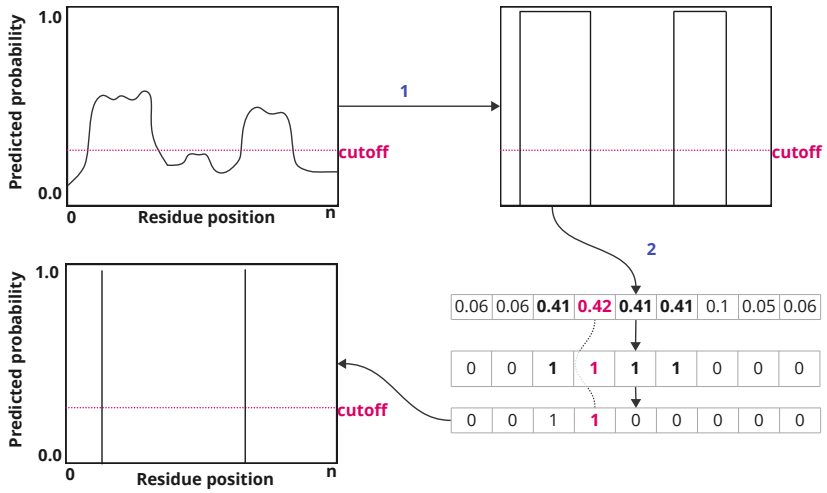


Figure 4.2: *Post-process methods. First, the probability vector is converted to a binary vector using a cutoff threshold. Then, the regions of boundaries will be removed by choosing the position in the region with the highest probability as the domain boundary.*

4.4 An overview of the machine learning system

So far, we have examined how the system will learn to predict domain boundaries. Figure 4.3 is a high-level abstract diagram that shows how the data flows in the system. We have discussed how the sequence is input to the system, the ground truth for each prediction, the choice of cut-off threshold and the problem of dealing with class imbalance.

[3]Learning a task through the transfer of knowledge from a related task. In this particular case is learning where the domain boundaries are from the predicted number of domain boundaries in a chain.

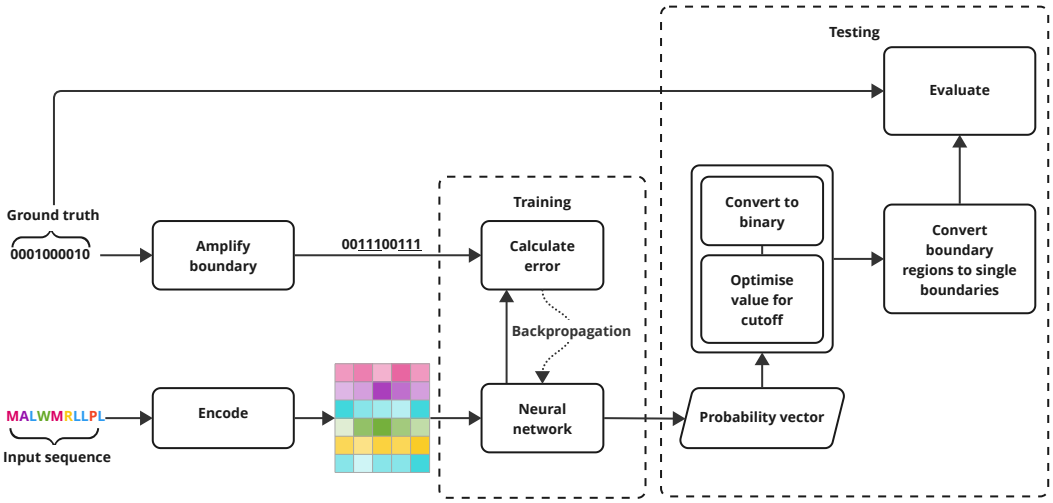


Figure 4.3: *Flowchart of the machine learning system design. Starting with the input sequence, it is given as input to the encoding mechanism that produces a representation matrix. During training the parameters of the neural network are optimised to minimise the error between the prediction and the amplified signal that comes from the ground truth. Finally, the output is post-processed to convert a probability vector to a binary vector.*

4.5 Neural network design

4.5.1 Main architecture

In Chapter 2 we looked at studies that employed recurrent neural networks, residual networks and convolutional neural networks.

Residual networks utilise skip connections which facilitates learning in much deeper networks. As we do not have the computational resources to train a very deep neural network we will not be considering this architecture as it will offer no benefit.

Recurrent neural networks perform very well among sequence-based tasks. Since domain boundary prediction involves learning which positions are boundaries from a sequence of amino acids, a recurrent neural network naturally fits this problem domain, which incolves prediction from sequence data. However, because amino acid sequences are typically between 50 and 2000 residues long (Alberts et al. 2002), classic recurrent neural networks may fail to learn such long-term dependencies and may train extremely slowly due to the gradient vanishing problem. Therefore, we employed the variant LSTM, which overcomes the limitations of the classic RNN.

Lastly, we considered using convolutional layers in the neural network architecture as well. However, we observed that using convolutional layers provided no benefit.

Therefore, we chose to use a bidirectional LSTM as the main component of the neural network.

4.5.2 Hyperparameter tuning

There is a plethora of variables in neural network and in order to find the optimal values for these variables we performed a linear optimisation[4]. A combinatoric[5] optimisation would have been

[4]Tuning hyperparameter A. Then tuning hyperparameter B using the same (best) value for hyperparameter A. Then tuning hyperparameter C using only the best values from A, B. All the way up to the last hyperparameter.

[5]Trying every single combination of possible configurations. For example, in order to tune 5 hyperparameters with 3 possible values for each, it would require 3*3*3*3*3 = 243 different configurations. Linear optimisation would only

Table 4.1: *The results from hyperparameter tuning. The hyperparameters were tuned with the order given in the table with the exception of the number of hidden features and number of FC layers which were tuned in conjunction as the number of hidden features affects the input size of the first FC layer. The values that yielded that worst and best MCC are shown along with the percentage increase from using one over the other.*

Order	Hyperparameter	Worst value	Optimal value	% Increase in MCC
1	Bidirectional	False	True	10.41
2	Optimizer	SGD (momentum=0.9)	Adam	61.49
3	# of hidden features	20	100	8.02
4	# of FC layers	1	6	8.02
5	# of convolutional layers	1	0	1.50
6	# of LSTM layers	1	3	3.63
7	Batch size	64	32	3.30
8	Activation function	Leaky ReLU	ReLU	4.58
9	Boundary region size	31	41	4.66

ideal, however, due to the number of different variables and possible values for each variable, we avoided this approach due to limited resources. We optimised the following variables in order:

1. Bidirectional (True/False)
2. Optimizer
3. Number of hidden LSTM features
4. Number of fully connected layers (FC) following the LSTM output
5. Number of convolutional layers prior to the LSTM
6. Number of LSTM (stacked) layers
7. Batch size
8. Activation function
9. Size of boundary regions during pre-processing

It is worth noting that we used MCC as the our guide during hyperparameter tuning. Moreover, we performed used ESM as the encoding mechanism. We avoided using CARP because it produces more features per residue compared to ESM (1280 vs 480), which would take longer to train on. Additionally, following a similar rationale we started training using one-hot which is only 21 features per residue. However, we ended up switching to ESM because one-hot was failing to yield an MCC higher than 0.

We kept constant:

- The encoding mechanism (ESM)
- The hardware used to train on (1x GPU Tesla T4 and Intel Core Xeon clocked at 2.30GHz)
- The training set and the order with which each data point was input to the model
- The learning rate (at 0.001)

Furthermore, it is worth noting that we started hyperparameter tuning using 3 epochs per training cycle. However, as we increased the number of LSTM layers, fully connected layers and number of hidden features, we started training using 5 epochs as the network required more iterations to start converging.

Table 4.1 shows the results from the process. We observe that the biggest increase in performance was due to making the LSTM bidirectional. The reason for this increase is because the neural network takes into consideration both the residues before and after the position it is currently

consider 15 configurations.

considering. This shows that both the information about the residues before and after each position is relevant in determining if a residue is a boundary or not.

4.5.3 Learning rate

We started with a learning rate of 0.001 which is the default learning rate for the Adam optimizer in PyTorch. We noticed that the training loss had a steep decrease within the first epoch but followed a more steady decline after epoch 1. The MCC followed a similar pattern. Figure 4.4 shows the values for the training loss and the MCC during training for 7 epochs. The model showed a sign of overfitting after the 6th epoch.

Moreover, we wanted to experiment with a learning rate scheduler which decreased the learning rate by a factor of 15% after every epoch. The effect of using a scheduler was the same as re-training the model with different starting weights, which suggests that the scheduler did not significantly impact the learning of the model.

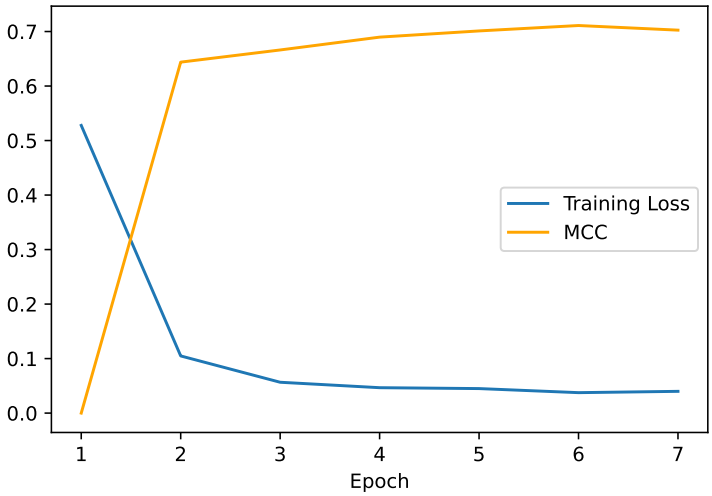


Figure 4.4: *The training loss and MCC plotted against epochs.*

4.5.4 The final architecture

After hyperparameter tuning, the neural network architecture is as follows:

- 3 stacked bidirectional LSTM layers with 100 hidden features.
- 6 fully connected layers with the following output sizes in order: 80, 60, 50, 20, 10, 1.
- ReLU activation after the last LSTM layer and every fully connected layer except the last.
- Sigmoid activation function after the last fully connected layer to convert the output to a probability.

Additionally, we used a batch size of 32 and a boundary region of length 41. The choice of optimizer was Adam with a learning rate of 0.001. See Figure 1 for the PyTorch implementation.

Table 4.2: *Distribution of sequences. First row shows the number of chains with n domains in CATH. Second row shows the number of sequences in our dataset after stratified sampling and removing homologous clusters using many-against-many sequence searching.*

Number of domains	1	2	3	4	5 or more
Number of sequences in CATH	**222,168**	**99,666**	**21,828**	**7,526**	**3,965**
Number of sequence in our dataset	**3,953**	**2,726**	**1,198**	**396**	**224**

4.6 Data collection and processing

4.6.1 Overview

To train the neural network we need to create a dataset with amino acid sequences of proteins coupled with their domain boundaries. We create this dataset using a combination of two of the publicly available databases discussed in Chapter 2: PDB and CATH. PDB provides `code.pdb` files for a large number of proteins, where each contains the amino acid sequence of the protein along with the 3D structure of the protein which has been determined using experimental methods. CATH provides a list of PDB codes coupled with their domain boundaries.

4.6.2 Sampling

We first started with sampling chains from the CATH list. We used stratified sampling to collect a random sample of chains of different sizes with regards to the number of domains. To do this, we parsed the entire list and put each chain into a category where each category is characterised by the number of domains the chains in the category have. Table 4.2 shows the distribution of sequences with *n* domains in CATH.

The number of domains a protein has is directly proportional to the number of its domain boundaries. Therefore, we created a balanced dataset in order to avoid biases from being introduced to our model.

4.6.3 Sequence length

We filtered out sequences of length greater than 2000. The reason is because in order to train our model in batches we need equally sized inputs and to do this we pad the sequences to be of the same length as the longest chain in our dataset. Limiting this length to 2000 also limits the computational resources required to train the model as it scales with the input size. Ideally we would train our model with sequences of greater length, however, we were limited by the available hardware. It is worth noting that the model can still take as input sequences of any length during testing which does not limit our solution from predicting the domain boundaries of any given chain as Shi et al. (2019) and Wang et al. (2022) do.

To remove chains longer than 2000 residues we first used a BASH script to bulk-download the `.pdb` file for each sequence we have sampled so far. Then, we use a Python library aimed at bioinformatics applications called Biopython (Biopython 2023), to extract the sequence of the chain using the `.pdb` file, which allows us to obtain the length of the sequence.

4.6.4 Sequence similarity

Many amino acid sequences are by nature homologous to many other sequences. Training on a dataset of homologous sequences can introduce a bias in the model as it is not exposed to enough

Table 4.3: *Rows 12 to 20 from the resulting file after many-to-many sequence searching using MMseqs showing a cluster of homologous sequences. It highlights that chain C in protein with PDB ID 3qb9 is homologous to 8 other different chains.*

Chain	Homologous to
3qb9:C	3qb9:C
3qb9:C	3uof:B
3qb9:C	3b3h:B
3qb9:C	5d8q:G
3qb9:C	4toa:S
3qb9:C	3e1p:B
3qb9:C	3e1l:D
3qb9:C	4tod:P
3qb9:C	3is7:G

variety of sequences. Furthermore, a model trained using homologous training and test sets will seem to perform better than it would if the test set shared no similarities with the training set.

4.6.5 Many-against-many sequence searching

After filtering out every sequence longer than 2000 residues, we collect the sampled sequences and perform many-again-many sequence searching to cluster chains into homologous clusters. Doing so allows to remove the clusters from our dataset, which in turn will remove homologous sequences in our dataset. Two sequences are homologous if they have at least 30% identical residues over 80% of the sequence. In other words, if 80% of two sequences align and they share 30% of the residues then they are considered homologous. Our dataset was initially 66219 chains long. This number dropped to 8497 after removing homologous clusters.

After stratified sampling and removing sequences that are too long, we use many-against-many sequence searching to cluster proteins into homologous clusters and remove every protein from the cluster except one. This will create a dataset without homogeneity. To do this, we used an open-sourced software called MMseqs (Hauser et al. 2016), which takes as input a text file with all the sequences we want to perform many-against-many search. The output from MMseqs is a `.tsv`[6] file with two columns (see Table 4.3). The left column contains the identifier of a chain and the right column contains the identifier of a chain that the chain on the left is homologous to.

4.6.6 Final dataset

After stratified sampling and many-against-many sequence searching, we are left with 8497 sequences. Table 4.2 shows the distribution of sequences with n domains in our dataset. Figure 4.6 provides a visualisation of the number of single and multi-domain chains along with a density plot of the sequence lengths for single and multi-domain sequences.

We created a more balanced dataset that makes it harder for biases to be introduced in our model. We also created an almost 50-50 split between single and multi-domain sequences, which is optimal for learning binary classification between single and multi-domain chains. The process with which we created the balanced dataset is illustrated in Figure 4.5.

[6] Tab separated values.

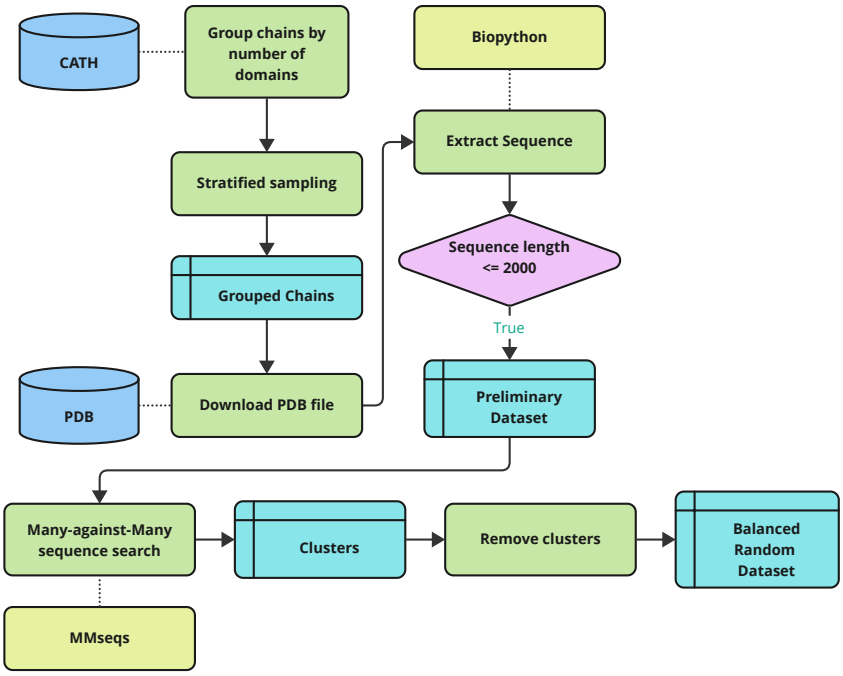


Figure 4.5: *Flowchart of the process to create a balanced and randomly sampled dataset using the publicly available databases CATH and PDB.*

4.7 CARP vs ESM

To compare the performance of our model using each of the encoding mechanisms, we need to maintain a controlled environment. To do this, we will control for all the variables during training and testing. That is, all the training hyperparameters, the neural network architecture and the order with which each data point in the training will tune the weights of the model will be constant.

Furthermore, we will perform 5-fold-cross validation to obtain a more robust estimate of the performance of each encoding mechanism. Finally, we will perform a statistical analysis by performing a t-test, in order to determine whether any difference in performance is statistically significant.

It is worth noting that the comparison will be made between versions of the models with a similar number of trainable parameters, to keep a more fair comparison. Our choice of CARP model will be `carp_38M` which has 38 million trainable parameters, and our choice of ESM model will be `esm2_t12_35M_UR50D` which has 35 million trainable parameters. Ideally, would aim to compare the state-of-the-art model for each, but, due to the enormous number of trainable parameters they have (over 600 million each) and the required computational resources to use, we will use smaller models.

4.8 AlphaFold evaluation pipeline

4.8.1 Overview

AlphaFold predicts the 3D structure of a protein from the amino acid sequence. In order to evaluate it in the domain boundary prediction task we must "translate" the 3D structure to protein

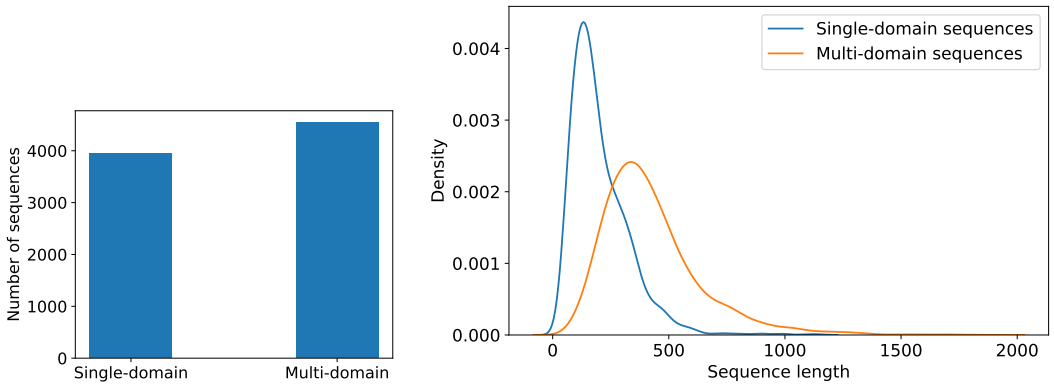


Figure 4.6: *Visualisation of our dataset after stratified sampling and many-against-many sequence searching. (a) shows the number of single and multi-domain chains. (b) shows the density of the lengths of sequences for single-domain and multi-domain sequences.*

domains. To do this, we use SWORD2 (Cretin et al. 2022), a more recent version of the method of Postic et al. (2017). Once a chain is assigned domains, we use the information to create binary vectors that indicate at which positions in the chain there exists a domain boundary, which is the same technique we employed when training our neural network model.

4.8.2 Domain assignment

The decision to use SWORD2 for domain assignment came from the fact the it provided the best documentation to use the tool. We also considered using DPAM (Zhang et al. 2023) which it was demonstrated during the evaluation phase that it could recognize 99.5% domains and assign correct boundaries for 85.2% of them. However, the documentation for the tool is very poor which motivated us to look at alternatives.

4.8.3 Creating a mapping

As described earlier, AlphaFold uses a different format for the unique identifier of proteins. We built a pipeline to create a one-to-one mapping between a PDB ID - the format CATH and PDB use to uniquely identify a protein - and a UniProt accession number - the format that AlphaFold uses to uniquely identify an entry in its database. To create the mapping we do the following:

- **Map the PDB ID to a UniProt accession**. We took every PDB ID from the chains in our dataset and mapped it to a UniProt accession number using UniProt's mapping tool[7]. The tool takes as input a list of PDB IDs and produces a JSON file from which we can deduce the UniProt accession number of the PDB ID through database cross-references, as well as if there exists an AlphaFold-predicted structure for the mapped UniProt accession.
- **Check if there exists an AlphaFold-predicted structure**. AlphaFold does not provide a prediction for every protein. Therefore, we filtered out the PDB IDs for which there does not exist a predicted structure.
- **Download structure**. We downloaded the AlphaFold-predicted structure, which is a `.pdb` file, from AlphaFold's database.

[7] www.uniprot.org/id-mapping

- **Extract the sequence**. After we downloaded the predicted structure, we used Biopython (Biopython 2023) to extract the amino acid sequence of the chain. We noticed that some chains could not be parsed correctly either because of a Biopython bug or due to the content of the `.pdb` file.
- **Similarity check**. The mapping between PDB ID and UniProt accession number is not always reliable, as we often noticed that a chain with a PDB ID was mapped to a UniProt accession number which had a different chain. Therefore we apply a similarity check between the two sequences to verify that they refer to the same, or very similar sequences. We initially considered the Needleman–Wunsch algorithm (Likic 2008) to perform the similarity check. However, noticing that a large amount of chains achieved a similarity score of 1.0, we started checking if two sequences are exactly identical which requires less resources and speeds up the process. If the UniProt chain did not exactly match the PDB chain, we filtered out the UniProt. If the they matched, we stored the two identifiers as a pair.

Figure 4.7 provides an illustration of this process.

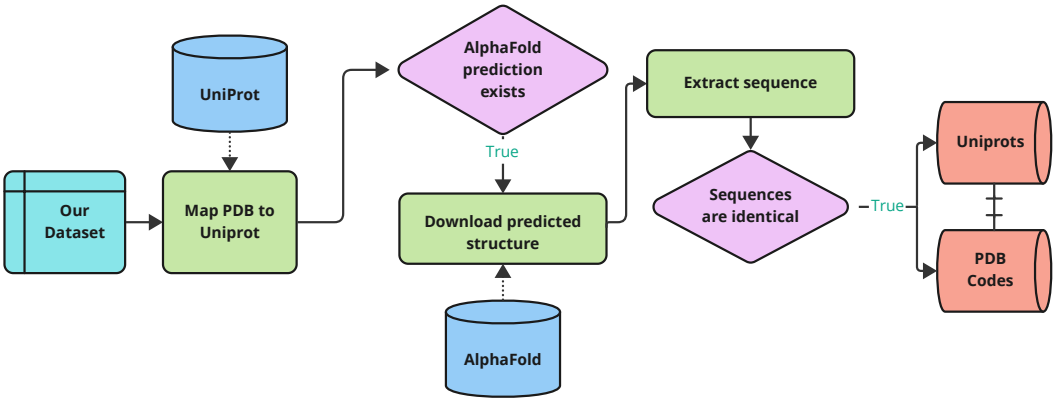


Figure 4.7: *Flowchart of the architecture and information flow of the reconciliation process between the pairs of PDB ID and UniProt accession numbers described in Section 4.8.3.*

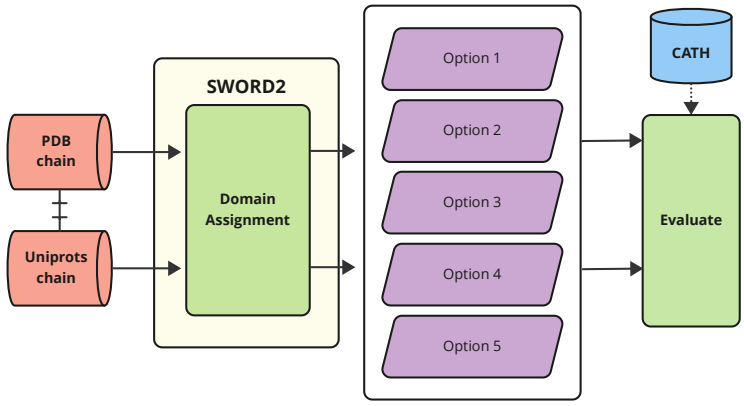


Figure 4.8: *Flowchart of the process of using the results from SWORD2 on AlphaFold to evaluate AlphaFold in domain boundary prediction.*

4.8.4 Evaluation method

SWORD2 takes as input a `.pdb` file and outputs a text file which contains one primary domain prediction, as well as 4 other alternative options. We parse the file, take each option, convert the domains to domain boundaries and compare that to the ground truth, which is the same ground truth as in Section 4.1.2, obtained from CATH. We take the score that the best option yields to be the score for a prediction. Figure 4.8 provides an illustration of the domain assignment and evaluation process.

In addition to evaluating the domain boundaries obtained from the AlphaFold predicted structure in combination with SWORD2, we also evaluate the domain boundaries obtained using the true `.pdb` file in conjunction with SWORD2. The results are particularly interesting as they show that the using SWORD2 on AlphaFold-predicted structures yields higher evaluation metrics than using SWORD2 on the experimental structures.

4.9 Summary

We described the methods we used to train a neural network a using amino acid sequences as input to predict domain boundaries. We looked at the pre-processing and post-processing methods that allow the neural network to accept an amino acid sequence as input and output a binary vector which indicates the positions in the sequence with domain boundaries. We described how we tuned the hyperparameters of the model using a linear optimisation and what the final architecture of the model looks like. Moreover, we looked at how we created a balanced and randomly sampled dataset that is used to train the neural network and how we took measures to avoid biases from being introduced during learning. Additionally, we described what the method to compare the three encoding mechanisms is. Finally, we looked at how AlphaFold is evaluated..

5 Evaluation

5.1 Evaluation metrics

As concluded by Postic et al. (2017), finding the correct number of domains is a more challenging problem than delimiting accurate boundaries, which provides an incentive to evaluate a model both on how well it delimits domains and how well it predicts the number of domains for a given protein. In order to evaluate the performance of our model and AlphaFold, we followed the convention in Zheng et al. (2020) and Wang et al. (2022), which use two different sets of metrics: domain number prediction metrics and boundary prediction metrics.

5.1.1 Domain number prediction

Domain number prediction metrics evaluate how well a model performs in classifying chains of proteins as single-domain or multi-domain. Single-domain proteins have only one domain, therefore, they have no domain boundaries. We used precision, recall, accuracy and the Mathew's Correlation Coefficient to evaluate the models in the domain number prediction task.

Mathew's Correlation Coefficient (MCC) is a metric used to evaluate model performance in the task of binary classification. Studies have shown that MCC is more reliable than other common metrics used in binary classification, such as accuracy and F-score (Chicco and Jurman 2020; Chicco et al. 2021), particularly when there exist a class imbalance. This has led to MCC being employed in the evaluation of models in the domain boundary prediction task.

5.1.2 Definition of metrics

Each of the aforementioned metrics is based on the confusion matrix generated by predicting each of the two domain classes. The confusion matrix counts the number of:

- true single-domain (TS) predictions
- false single-domain (FS) predictions
- true multi-domain (TM) predictions
- false multi-domain (FM) predictions

Precision and recall are defined by:

$$Precision(Single) = \frac{TS}{TS + FS}, \quad Precision(Multi) = \frac{TM}{TM + FM},$$

$$Recall(Single) = \frac{TS}{TS + FM}, \quad Recall(Multi) = \frac{TM}{TM + FS}.$$

Table 5.1 describes what the two metrics describe with regards to domain number prediction. In general, precision measures how many retrieved items are relevant, and recall measures how many relevant items are retrieved.

Accuracy is defined by:

$$Accuracy = \frac{TM + TS}{TM + TS + FM + FS},$$

Table 5.1: *A table presenting an intuitive explanation of precision and recall as domain number prediction metrics. The questions in the four cells highlight the question that the metric (row) provides an answer for.*

	Single-domain	Multi-domain
Precision	How many proteins classified as single-domain are indeed single-domain?	How many proteins classified as multi-domain are indeed multi-domain?
Recall	Out of all the true single-domain proteins, how many were correctly classified as single-domain?	Out of all the true multi-domain proteins, how many were correctly classified as multi-domain?

which tells us how many times a prediction was correct overall.

Finally, MCC is defined by:

$$MCC = \frac{(TM + TS) \times (FM + FS)}{\sqrt{(TM + FM) \times (TM + FS) \times (FM + TS) \times (TS + FS)}}.$$

The MCC is a special case of the Pearson Correlation Coefficient (Cohen et al. 2009) which is a bivariate correlation. Akoglu (2018) provides a guide to interpreting the Pearson Correlation Coefficient, which we also employed during the evaluation of our model.

5.1.3 Boundary prediction

To evaluate the domain delimitation of our model we used the domain boundary distance (DBD) score as defined in Tress et al. (2007). DBD measures how close the predicted boundaries are from the true boundaries. We consider a predicted boundary within ±8 residues from the true boundary to be a true positive. Figure 2 shows how we implemented this metric.

5.2 Comparing ESM with CARP

We performed 5-fold-cross-validation and obtained a mean value for each of the metrics we use to evaluate the performance of the model. Table 5.3 presents the results. In order to determine if the difference in the values between each of the models trained with a different encoding mechanism, we performed a t-test between the DBD and MCC scores of each model trained with a different encoding mechanism. The reason we use DBD and MCC is because each provides information regarding the performance of the model in domain number prediction and boundary prediction respectively. This gives a more complete picture of how each of the models performs. In addition, we chose MCC over the rest of the metrics because, as discussed in Section 5.1.1, MCC is considered a more reliable metric than alternatives, such as accuracy, in measuring the bivariate relationship between predictions and the ground truth in classifying chains as single-domain or multi-domains. Table 5.2 shows the results of the t-tests.

5.2.1 Training and encoding times

We measure the time taken for the neural network to train using each of the encoding methods. We also measured the time taken when using one-hot encoding as the baseline. In addition, we measure the time taken to generate the matrix representations of our dataset using each of the methods. Table 5.4 shows the results we obtained.

Table 5.2: *Statistical analysis (t-test) of the performance of each model trained with a different encoding mechanism. Note that, the p-value between the mean DBD scores of the pairs (ESM, one-hot) and (CARP, one-hot) were so small that an underflow error occurred and returned the value of 0.*

Pair	MCC		DBD	
	t-statistic	p-value	t-statistic	p-value
ESM, CARP	1.6266	$1.4246 \cdot 10^{-1}$	23.4	$1.6945 \cdot 10^{-118}$
ESM, one-hot	33.2153	$7.3658 \cdot 10^{-10}$	68.3	0
CARP, one-hot	19.4431	$5.0856 \cdot 10^{-8}$	52.4	0

Table 5.3: *Comparison of the prediction performance of our model using 5-fold-cross-validation using the three different encoding methods discussed in Chapter 2 sorted by the MCC score.*

Methods	Domain number prediction						Boundary prediction
	Single-domain		Multi-domain		All		
	Pre	Rec	Pre	Rec	Acc	MCC	DBD
ESM	0.9244	0.8104	0.7392	0.9452	0.8494	0.7088	0.5596
CARP	0.9009	0.7768	0.6819	0.9328	0.8175	0.6446	0.4529
One-hot	0.4643	0	1	0	0.4642	0	0.4643

Table 5.4: *The time taken for to train using each method and time taken to encode the 8497 chains in our dataset using each method. To obtain the training times, we trained the model for 5 epochs on 80% of the 8497 chains and took the mean number of seconds taken. To obtain the encoding times, we encoded the chains 3 times and took the mean number of seconds taken. We used a NVIDIA A100-SXM4-40GB GPU for and Intel(R) Xeon(R) CPU @ 2.20GHz CPU for encoding. We used a GPU Tesla T4 GPU and Intel(R) Xeon(R) CPU @ 2.20GHz CPU for training our model.*

Methods	Training time	Encoding time
ESM	257 seconds	203 seconds
CARP	429 seconds	168 seconds
One-hot	115 seconds	17 seconds

Table 5.5: *Comparison of the prediction performance of our model and the three latest methods described in Chapter 2 sorted by the MCC score.*

Methods	Domain number prediction						Boundary prediction
	Single-domain		Multi-domain		All		
	Pre	Rec	Pre	Rec	Acc	MCC	DBD
Res-Dom	0.963	0.788	0.667	0.933	0.833	0.674	0.532
Our model	0.865	0.679	0.833	0.731	0.800	0.554	0.125
FUpred	0.95	0.576	0.500	0.933	0.688	0.479	0.578
DNN-Dom	0.839	0.788	0.588	0.667	0.750	0.441	0.457

Table 5.6: *Comparison of the prediction performance of AlphaFold with SWORD2 and PDB (the experimental structure) with SWORD2, tested on 503 chains.*

Methods	Domain number prediction						Boundary prediction
	Single-domain		Multi-domain		All		
	Pre	Rec	Pre	Rec	Acc	MCC	DBD
AlphaFold / SWORD2	1	0.8960	0.9483	1	0.9642	0.9217	0.3173
PDB / SWORD2	1	0.8613	0.9322	1	0.9523	0.8960	0.1688

5.3 Testing on independent data

In order to be able to compare our model with the methods in Chapter 2, we tested our model on the independent dataset which the other methods were tested on as well. The dataset is the CASP13[1] dataset which was used to test state-of-the-art methods in 2018. This dataset includes 80 chains with domain numbers ranging from 1 to 7 domains. Table 5.5 presents the results when we tested our model on CASP13.

5.4 AlphaFold evaluation

We evaluated AlphaFold in combination with SWORD2 in domain boundary prediction. In addition, we evaluated SWORD2 in combination with the true, experimental 3D structures of the same chains. Table 5.6 presents the results obtained from this evaluation.

5.5 How good is our solution overall?

Our model achieved an MCC of 0.554 in classifying proteins as single-domain or multi-domain. This is the second highest MCC out of the methods we looked at in Chapter 2. According to Akoglu (2018), this value for an MCC suggests a strong relationship between the ground truth and the predictions. It is important to note that this classification is implicit, rather than explicit. That is, the model does explicitly predict the label of chains. We infer the classification of the chain from the number of residues that were predicted to be domain boundaries. Moreover, our model achieved a DBD score of 0.125, which is the lowest score seen compared to the methods

[1] https://predictioncenter.org/casp13/

we looked at in Chapter 2. This suggests that our model performs poorly in predicting the precise position of boundaries in a sequence.

5.6 How well does our solution identify single-domain and multi-domain proteins?

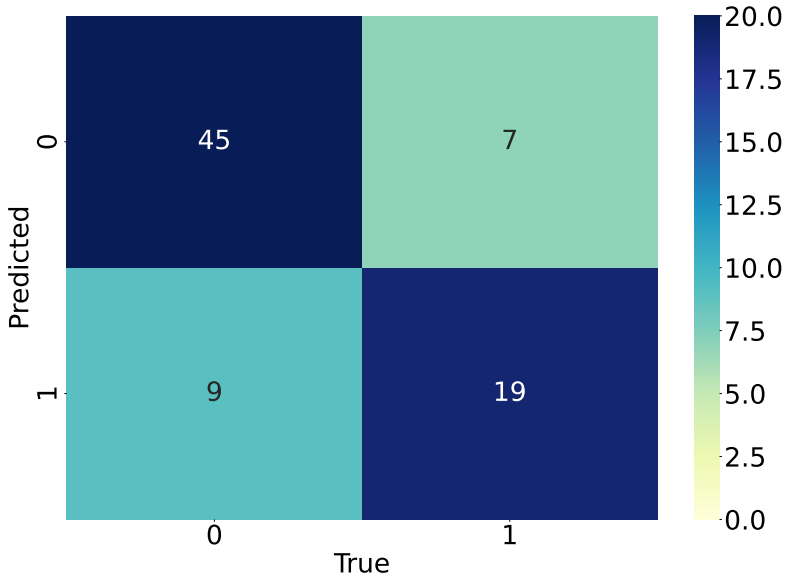
Our model, overall, obtains a higher precision score than recall score, which suggests that our model is better at identifying true positives (single-domain) but may miss some actual true positives by making false-negative (multi-domain) predictions. In other words, our model is more likely to classify a single-domain chain as single-domain if it is confident in the prediction. The same applies for multi-domain chains as the precision score for each class is higher than the recall of each class. Looking at Figure 5.1a, we observe that that the top-left to bottom-right diagonal is highlighted with more intensity which also suggests that strong bivariate relationship between the ground truth and predictions. It is important to note that, the precision and recall metrics are obtained after optimising a cutoff threshold t to maximise the MCC score. We discussed in Section 4.3.2 how using different values for t would result in different values for different metrics. Therefore, it would be possible to adjust the true/false positive/negative rate of the predictions by adjusting this cutoff threshold t.

5.7 How well does our solution predict the exact number of domains?

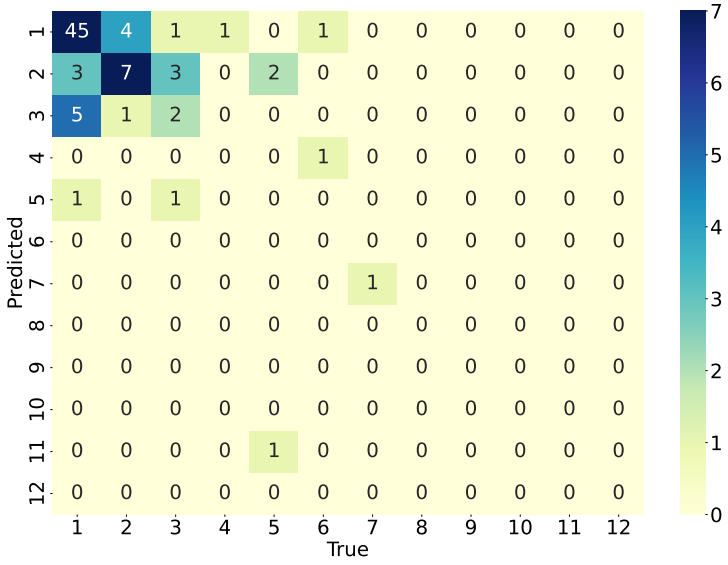
Looking at Figure 5.1b we see that there is no apparent relationship between the true and predicted number of domains. This suggests that our solution fails to accurately predict the exact number of domains of a chain. In addition, we observe that the model does not tend to either over-predict nor under-predict the number of domains, as the number of items below the top-left to bottom-right diagonal is very close to the number above it (12 and 13 respectively). However, the cases where the model over-predicted the number are more extreme. For example, in the case where the true number of domains for a chain is 5, the model predicted 11 domains. This is particularly interesting and raised the question if this has to do with the number of residues in the chain.

5.8 Is there a relationship between the sequence length and the number of predicted domains?

We plotted the number of domains against the number of residues for each chain. Figure 5.2a plots the predicted number of domains and Figure 5.2b plots the true number of domains. We make an interesting observation that the predictions are more regular than nature, as the the correlation coefficient for the points in Figure 5.2a is 0.8142, and the correlation coefficient for points in Figure 5.2b is 0.4908, which proves that there is a strong correlation between the predicted number and the sequence length and a much weaker correlation between the true number and the sequence length. Moreover, we observe that in the case extreme case we looked at earlier where the true number of domains is 5 and the model predicted 11, the sequence length is 1600 residues long which is a much longer chain than the rest in the dataset. This observation suggests that our model may implicitly be using the sequence length as information when making predictions. Looking at Figure 5.3, we see that this relationship exists in the dataset we used to train our model, as the sequence length and domain number have a correlation coefficient of 0.7308.

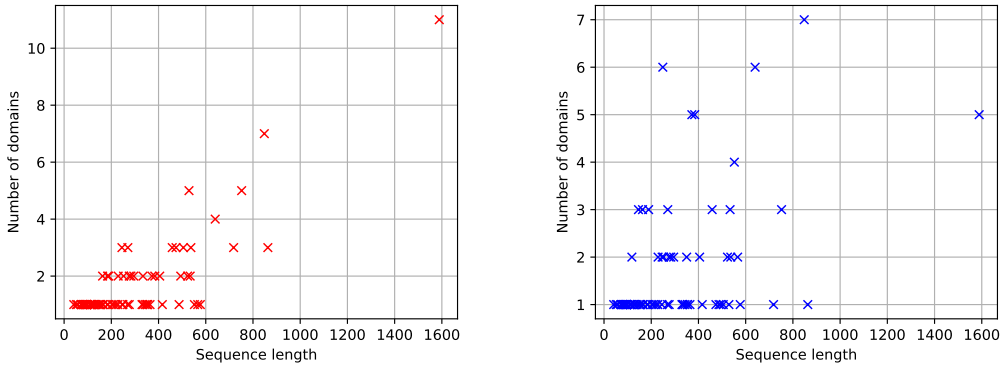


(a) *Single/Multi domain confusion matrix. A prediction of 0 indicates a single-domain chain. A prediction of 1 indicates a multi-domain chain*



(b) *Confusion matrix of the number of true and predicted domains.*

Figure 5.1: *Confusion matrices of the predictions of the model. (a) shows the confusion matrix of single-domain and multi-domain predictions. (b) shows the confusion matrix of the domain number prediction.*



(a) Predicted domain number

(b) True domain number

Figure 5.2: *Number of domains of chains in the CASP13 dataset against the sequence length. (a) shows the relationship using the predicted domain number and (b) shows the relationship using the true domain number. The correlation coefficient of (a) is 0.8142. The correlation coefficient of (b) is 0.4908.*

5.9 Which encoding mechanism helped our model make the best predictions?

Looking at Table 5.3 we see that when using ESM as the encoding mechanism yields the highest MCC and DBD scores. After testing if the results are statistically significant, we observe that the difference in DBD is statistically significant. In addition we observe that the difference in MCC between ESM and one-hot encoding is also statistically significant. However, the difference in DBD between ESM and CARP is not statistically significant, as we obtained a p-value > 0.05, which allows us to reject the hypothesis that using ESM yields a higher MCC score than when using CARP, which is the convention used in hypothesis testing. Overall, we can conclude that we are confident that when using ESM our model yields a better MCC score, but the difference in DBD could be noise. Furthermore, it is important to note that, because ESM produces only 480 features per residue, whereas CARP produces 1280 features per residue, our model trained faster when using ESM. However, as ESM is based on the Transformer architecture which is less efficient than the convolutional autoencoder architecture that CARP is based on, the time taken to encode our amino acid sequences when using ESM is greater than the time taken to encode using CARP. Table 5.4 presents the times that our model took to train using each of the encoding mechanisms, and the time taken to encode our dataset using each method.

5.10 How does AlphaFold perform in domain boundary prediction?

Looking at Table 5.6 we observe that using SWORD2 on the AlphaFold-predicted structures performs very well when classifying chains as single-domain and multi-domain. However, the precise predicted position of domain boundaries is not as good as the state-of-the-art methods used in domain boundary prediction, as the DBD score is low. However, we observe that the DBD when using SWORD2 in conjunction with the true, experimental protein structures, also yields a low DBD score. This suggests that the inaccurate prediction of the precise position of boundaries may be a limitation of SWORD2, which was used to assign domains to the 3D structures of the proteins. In addition, we make an interesting observation that, when using

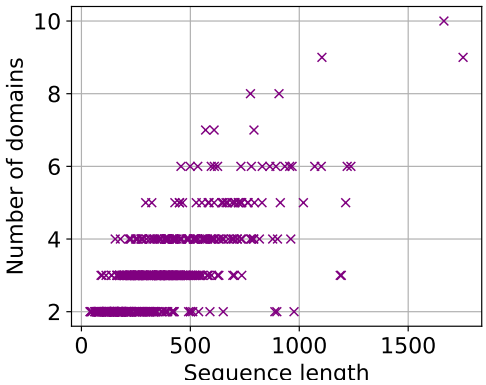


Figure 5.3: *Number of domains of chains in our dataset (the dataset our model was trained on) against the sequence length. Correlation coefficient: 0.7308.*

SWORD2 in conjunction with AlphaFold yields higher metric scores than when using SWORD2 in conjunction with the experimental structures. Understanding why this is the case would require further testing and a better look at the algorithms used by SWORD2. Therefore, we consider this to be part of the future work. Overall, due to the limitation of SWORD2, we cannot determine if AlphaFold accurately identifies the precise position of domain boundaries, but we observe that it performs well in classifying domain boundaries.

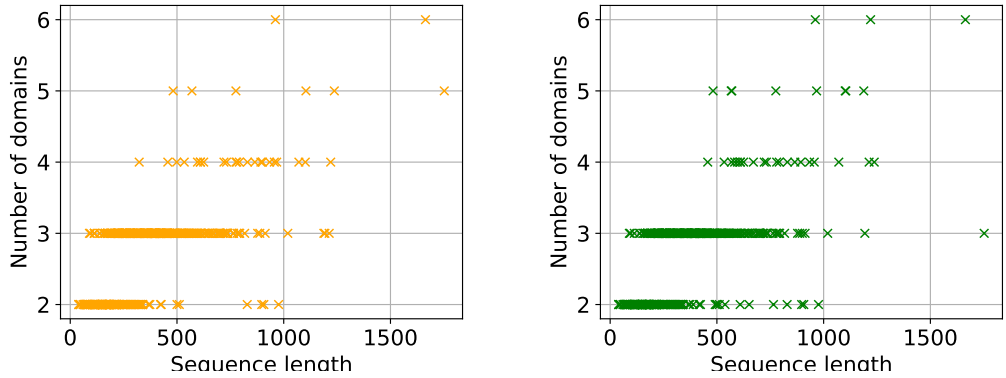
Furthermore, looking at Figure 5.4, we observe that there exists a relationship between the predicted domain number and the sequence length when using either PDB with SWORD2 or AlphaFold with SWORD2. However, the relationship is not as strong as the relationship in the true relationship seen in Figure 5.3. Lastly, according to Figure 5.5, the combination of AlphaFold and SWORD2, tends to underestimate the number of domains in the chains, as most points in the confusion matrix are above the top-left to bottom-right diagonal line.

5.11 What are the limitations of our evaluation methods?

One limitation of our method that stands out is that the hyperparameter phase in our process was completed while using only ESM to encode the amino acid sequences. It is therefore possible that there exist hyperparameter values that when coupled with CARP as the encoding mechanism, would yield higher metric scores than the ones we obtained. However, due to limited time and computational resources, we chose to perform hyperparameter tuning while using ESM as it was much faster for our model to train than when we used CARP.

Furthermore, the performance of AlphaFold in domain boundary prediction is only as good as the domain assignment tool we are using. Earlier we discussed how the low DBD scores obtained by AlphaFold may be a limitation of SWORD2 and not AlphaFold. Ideally, we would use more accurate methods but due to limited documentation on how to use and build them locally, we chose SWORD2 as our domain assignment method.

Finally, a limitation when comparing our method with the methods in literature is that the implementation of the DBD evaluation metric may have been different than the implementation used for state-of-the-art methods. This is because the evaluation metric is specific to this problem domain and therefore a well-defined algorithm does not exist. There is, however, a description of the steps taken to calculate the DBD score in Tress et al. (2007), but there are some ambiguities that may affect the overall score.



(a) *PDB/SWORD - Domain number*

(b) *AlphaFold/SWORD - Domain number*

Figure 5.4: *Number of domains of chains against the sequence length. (a) shows the relationship using the number obtained using PDB in conjunction with SWORD2. (b) shows the relationship using the number obtained using AlphaFold in conjunction with SWORD2. The correlation coefficient of (a) is 0.6508. The correlation coefficient of (b) is 0.6287.*

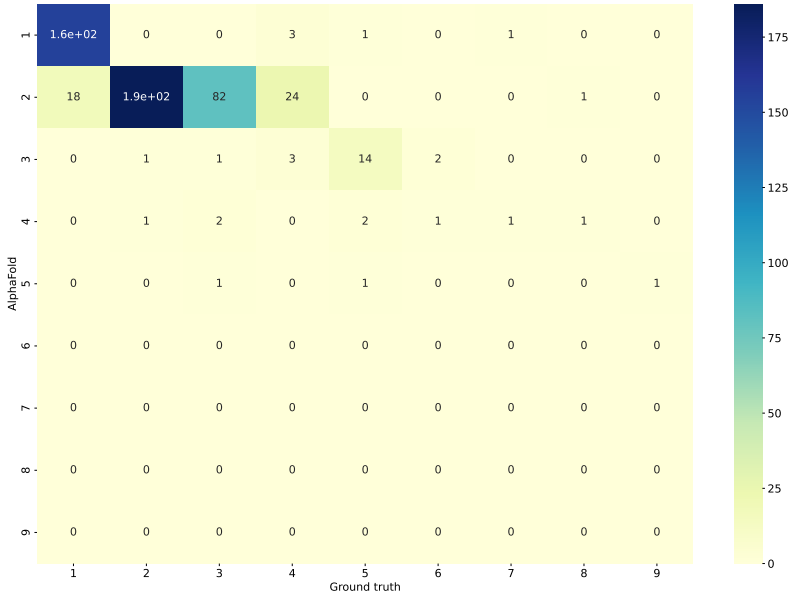


Figure 5.5: *Confusion matrix with the predicted number of domains predicted using AlphaFold/SWORD2 (y-axis) and the true number of domains (x-axis).*

6 Conclusion

6.1 Summary

The project explored the use of deep learning to predict domain boundaries in protein chains. A system was implemented to overcome the limitation of input sequences under a particular length, by relying only on the generation of matrix representations of amino acid sequences by pre-trained language models, whereas methods in the literature relied on methods to extract features from sequences that had a limit on the length of the sequences they accept. The solution we created was effective at classifying protein chains as multi-domain or single-domain, but not accurate in identifying precise boundary positions. The model was trained using two state-of-the-art protein language models (ESM and CARP), and the results showed that ESM achieved a higher MCC score for classification than CARP. However, there was no statistically significant difference in their DBD score, which evaluated how close the predicted boundaries were to the true boundaries. Additionally, the project evaluated AlphaFold in domain boundary prediction and concluded that the method used to assign domains (SWORD2) to the AlphaFold predicted structures was a limitation in our method, as SWORD2 in combination with the experimental protein structures yield a very low DBD score. However, AlphaFold in conjunction with the domain assignment method yield a very high MCC score, which suggests that AlphaFold does well in classifying chains as single-domain or multi-domain.

6.2 Future work

Moving forward, there are several areas that we could like to explore in order to gain a better understanding at how different neural network architecture and methods perform in the protein domain boundary prediction task.

To start with, we would like to experiment with the U-Net architecture to segment amino acid sequence into boundary and non-boundary regions. U-Nets are fully convolutional neural networks that have shown excellent performance in image segmentation tasks (Ronneberger et al. 2015) and may also be a suitable choice for predicting domain boundaries as well.

Another avenue for future work is to explore transfer learning, where the model would first learn the number of domains in each chain and then use that information to predict the precise positions of domain boundaries. This approach could potentially improve the accuracy of domain boundary prediction, as we have seen that our method does not perform well in predicting the exact number of domains in a chain.

Furthermore, we would like to investigate the fine-tuning of the pre-trained language models we utilised in our methods. To do this, we could create an end-to-end learning system where the weights of both the protein language model and the model to predict domain boundaries are tuned during learning. We would be interested to explore if this approach has the potential to improve the accuracy of the prediction model by allowing the language model to produce more meaningful and informative matrix representations for each amino acid sequence.

Finally, ideally we want to use a more accurate domain assignment method in conjunction with AlphaFold, to gain a better understanding of AlphaFold's ability to delimit domains.

Amino acids

Table 1: *A Table with the 20 amino acids needed to make all the proteins found in the human body and most other forms of life. Note that in rare cases proteins have amino acids not listed in the table. In those cases, the letter 'X' is used for the single letter abbreviation. Furthermore, sometimes it is very difficult or impossible to differentiate two closely related amino acids such as Glutamine and Glutamic acid. In that case, the amino acid is abbreviated as Glx or 'Z'*

Amino acid	Abbreviation	Single letter abbreviation
Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cysteine	Cys	C
Glutamine	Gln	Q
Glutamic acid	Glu	E
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V
Glutamine/Glutamic acid	Glx	Z
Other amino acids	-	X

PyTorch code of our models

Figure 1 shows the PyTorch code we used to implement the bi-directional LSTM.

DBD implementation in Python

Figure 2 shows the Python code we used to implement the domain boundary distance score.

```
class BiLSTM(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers):
        super(BiLSTM, self).__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers,
        ↪  batch_first=True, bidirectional=True, dropout=0.0)
        self.fc0 = nn.Linear(hidden_size*2,80)
        self.fc1 = nn.Linear(80,60)
        self.fc2 = nn.Linear(60,50)
        self.fc3 = nn.Linear(50,20)
        self.fc4 = nn.Linear(20,10)
        self.fc5 = nn.Linear(10, 1)
        self.activation = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        h0 = torch.zeros(self.num_layers*2, x.size(0),
        ↪  self.hidden_size).to(device)
        c0 = torch.zeros(self.num_layers*2, x.size(0),
        ↪  self.hidden_size).to(device)
        out, _ = self.lstm(x, (h0, c0))
        out = self.activation(out)
        out = self.fc0(out)
        out = self.activation(out)
        out = self.fc1(out)
        out = self.activation(out)
        out = self.fc2(out)
        out = self.activation(out)
        out = self.fc3(out)
        out = self.activation(out)
        out = self.fc4(out)
        out = self.activation(out)
        out = self.fc5(out)
        out = self.sigmoid(out)
        return out
```

Figure 1: *Code for the Bi-LSTM implemented in PyTorch.*

```
def dbd_score(y_pred, y_true, margin=8):
scores = []
number_of_true_boundaries = np.sum(y_true)
number_of_pred_boundaries = np.sum(y_pred)
denominator = max(number_of_true_boundaries,number_of_pred_boundaries)

if denominator == 0:
    return 'n/a'

y_true_cp = np.copy(y_true)
for i in range(len(y_pred)):
    window = y_true_cp[max(0, i-margin):min(len(y_true_cp), i+margin+1)]
    indices_window = list(range(max(0, i-margin), min(len(y_true_cp),
    ↪  i+margin+1)))
    if y_pred[i] == 1.0:
        if 1.0 in window:
            positions_with_boundaries = np.argwhere(window == 1).flatten()
            js = [indices_window[pos] for pos in positions_with_boundaries]
            closest_j = js[0]
            for j in js:
                if abs(j - i) < abs(closest_j - i):
                    closest_j = j
            diff = abs(i - closest_j)
            score = ((margin - diff) + 1) / (margin + 1)
            y_true_cp[closest_j] = 0
            scores.append(score)
return np.sum(scores) / denominator
```

Figure 2: *Python implementation of DBD.*

Bibliography

Akoglu, H. (2018), 'User's guide to correlation coefficients', *Turkish journal of emergency medicine* **18**(3), 91–93.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. and Walter, P. (2002), The shape and structure of proteins, *in* 'Molecular Biology of the Cell. 4th edition', Garland Science.

Anfinsen, C. B. (1973), 'Principles that govern the folding of protein chains', *Science* **181**(4096), 223–230.
URL: *https://www.science.org/doi/abs/10.1126/science.181.4096.223*

Bank, D., Koenigstein, N. and Giryes, R. (2020), 'Autoencoders', *arXiv preprint arXiv:2003.05991* .

Biopython (2023), 'Biopython', `https://github.com/biopython/biopython`. Last accessed: 2023-03-09.

Cheng, J., Randall, A. Z., Sweredoski, M. J. and Baldi, P. (2005), 'SCRATCH: a protein structure and structural feature prediction server', *Nucleic Acids Research* **33**(suppl$_2$), $W72 - -W76$.
URL:*https://doi.org/10.1093/nar/gki396*

Chicco, D. and Jurman, G. (2020), 'The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation', *BMC genomics* **21**, 1–13.

Chicco, D., Tötsch, N. and Jurman, G. (2021), 'The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation', *BioData mining* **14**(1), 1–22.

Cohen, I., Huang, Y., Chen, J., Benesty, J., Benesty, J., Chen, J., Huang, Y. and Cohen, I. (2009), 'Pearson correlation coefficient', *Noise reduction in speech processing* pp. 1–4.

Cretin, G., Galochkina, T., Vander Meersche, Y., de Brevern, A., Postic, G. and Gelly, J.-C. (2022), 'SWORD2: hierarchical analysis of protein 3D structures', *Nucleic Acids Research* **50**(W1), W732–W738.
URL: *https://doi.org/10.1093/nar/gkac370*

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018), 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805* .

Hauser, M., Steinegger, M. and Söding, J. (2016), 'Mmseqs software suite for fast and deep clustering and searching of large protein sequence sets', *Bioinformatics* **32**(9), 1323–1330.

Hochreiter, S. and Schmidhuber, J. (1996), Lstm can solve hard long time lag problems, *in* 'Proceedings of the 9th International Conference on Neural Information Processing Systems', NIPS'96, MIT Press, Cambridge, MA, USA, p. 473–479.

Jiang, Y., Wang, D. and Xu, D. (2019), 'DeepDom: Predicting protein domain boundary from sequence alone using stacked bidirectional LSTM', *Pac. Symp. Biocomput.* **24**, 66–75.
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6417825/*

Jumper, J., Evans, R., Pritzel, A. and et al. (2021), 'Highly accurate protein structure prediction with alphafold', *Nature* **596**(7873), 583–589.

Likic, V. (2008), 'The needleman-wunsch algorithm for sequence alignment', *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne* pp. 1–46.

Lopez, M. J. and Mohiuddin, S. S. (2022), *Biochemistry, Essential Amino Acids*, StatPearls Publishing.
URL: *http://www.ncbi.nlm.nih.gov/books/NBK557845/*

Pereira, J., Simpkin, A. J., Hartmann, M. D., Rigden, D. J., Keegan, R. M. and Lupas, A. N. (2021), 'High-accuracy protein structure prediction in casp14', *Proteins: Structure, Function, and Bioinformatics* **89**(12), 1687–1699.

Postic, G., Ghouzam, Y., Chebrek, R. and Gelly, J.-C. (2017), 'An ambiguity principle for assigning protein structural domains', *Science Advances* **3**(1), e1600552.
URL: *https://www.science.org/doi/abs/10.1126/sciadv.1600552*

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J. et al. (2021), 'Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences', *Proceedings of the National Academy of Sciences* **118**(15), e2016239118.

Ronneberger, O., Fischer, P. and Brox, T. (2015), U-net: Convolutional networks for biomedical image segmentation, *in* 'Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18', Springer, pp. 234–241.

Sampaleanu, L. M., Vallée, F., Thompson, G. D. and Howell, P. L. (2001), 'Three-dimensional structure of the argininosuccinate lyase frequently complementing allele q286r', *Biochemistry* **40**(51), 15570–15580.

Shi, Q., Chen, W., Huang, S., Jin, F., Dong, Y., Wang, Y. and Xue, Z. (2019), 'DNN-Dom: predicting protein domain boundary from sequence alone by deep neural network', *Bioinformatics* **35**(24), 5128–5136.
URL: *https://doi.org/10.1093/bioinformatics/btz464*

Tress, M., Cheng, J., Baldi, P., Joo, K., Lee, J., Seo, J.-H., Lee, J., Baker, D., Chivian, D., Kim, D. et al. (2007), 'Assessment of predictions submitted for the casp7 domain prediction category', *Proteins: Structure, Function, and Bioinformatics* **69**(S8), 137–151.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017), 'Attention is all you need', p. 6000–6010.
URL: *https://dl.acm.org/doi/10.5555/3295222.3295349*

Wang, L., Zhong, H., Xue, Z. and Wang, Y. (2022), 'Res-Dom: predicting protein domain boundary from sequence using deep residual network and Bi-LSTM', *Bioinformatics Advances* **2**(1). vbac060.
URL: *https://doi.org/10.1093/bioadv/vbac060*

Watson, B. A. (2021), 'Picture-perfect proteins', *Pegasus The Magazine of the University of Central Florida* .
URL: *https://www.ucf.edu/pegasus/picture-perfect-proteins/*

Yang, K. K., Lu, A. X. and Fusi, N. (2022), 'Convolutions are competitive with transformers for protein sequence pretraining', *bioRxiv* pp. 2022–05.

Zhang, J., Schaeffer, R. D., Durham, J., Cong, Q. and Grishin, N. V. (2023), 'Dpam: A domain parser for alphafold models', *Protein Science* **32**(2), e4548.

Zheng, W., Zhou, X., Wuyun, Q., Pearce, R., Li, Y. and Zhang, Y. (2020), 'FUpred: detecting protein domains through deep-learning-based contact map prediction', *Bioinformatics* **36**(12), 3749–3757.